Distribution Based Data Filtering for Financial Time Series Forecasting

Goce Ristanoski[1], James Bailey[1]

[1] The University of Melbourne, Melbourne, Australia
g.ristanoski@pgrad.unimelb.edu.au, baileyj@unimelb.edu.au

Abstract. Changes in the distribution of financial time series, particularly stock market prices, can happen at a very high frequency. Such changes make the prediction of future behavior very challenging. Application of traditional regression algorithms in this scenario is based on the assumption that all data samples are equally important for model building. Our work examines the use of an alternative data pre-processing approach, whereby knowledge of distribution changes is used to pre-filter the training dataset. Experimental results indicate that this simple and efficient technique can produce effective results and obtain improvements in prediction accuracy when used in conjunction with a range of forecasting techniques.

Keywords: Time series classification, regression, distribution change.

1 Introduction

Prediction techniques for the behavior of financial time series have been intensively studied [1][2]. A prime example is the forecasting of stock prices, which aims to forecast the future values of the price of a stock, in order to obtain information about its trends and direction of movement and thus allow the development of buying/selling strategies to gain competitive advantage.

Classic and popular methods for stock price forecasting [3][4] for both univariate and multivariate time series data include linear regression, hidden markov models, neural networks [11] and support vector machines [7].

The underlying data for financial time series may span a frequency as small as hourly or as long as several years. The longer the time interval, the more likely it is that the data samples will not follow the same distribution [8]. The classic statistical [13] and data mining time series prediction methods [14], at least in their simple form, do not take into consideration that such changes in distribution over time may occur with financial time series data. This can lead to a loss in prediction accuracy, since the prediction model that is built places equal value on all samples, even those whose distribution is not close to the distribution of the samples in the most recent past.

In this paper, we address the challenge of forecasting the behavior of time series using distribution change. In particular, we propose a technique for filtering the samples in such time series, in order to project out those samples which appear least relevant and retain those samples which appear most relevant for prediction. Our proposed **Distribution Based Samples Removing (DBSR)** algorithm operates by i) initially analyzing the time series to determine its different distributions, and then ii) reducing the time series by filtering out the samples whose distribution is furthest from the recent past. We develop two versions of the algorithm, one parametric and the other non-parametric. Our approach is designed to work for regression with univariate series that use a five day relative difference in percentage of price (RDP) format [16], but the approach can also be applied to original univariate time regressed on itself, as well as multivariate time series.

Our proposed data filtering method has a number of desirable properties: i) it is clean, simple and intuitive, ii) it is easy to implement and runs efficiently, since it is a data pre-processing step and thus iii) it can be used in conjunction with many existing time series prediction methods. Finally, we find that iv) it can help obtain improvements in prediction performance when used as a prior step to produce input for classic time series prediction algorithms.

2 Related work

There is a large amount of literature dealing with classification and regression for financial time series. Descriptions of classic methods can be found in standard textbooks such as [1][2][3]. Instead we briefly review related work that can be used for dataset filtering or pre-processing, since this is an essential feature of our approach.

Selecting samples from a set can be performed by simple random sampling, cluster sampling, systematic sampling, or load shedding [5], but most of these methods do not consider the time element that is present when dealing with financial time series. Efforts have been by [21][22] to improve these methods and to include the time element, by using strategies based on sliding windows [22]. Nevertheless, sample selection in time series mostly consists of only selecting a continuous sample set, without the possibly of removing non contiguous ranges of samples from the set.

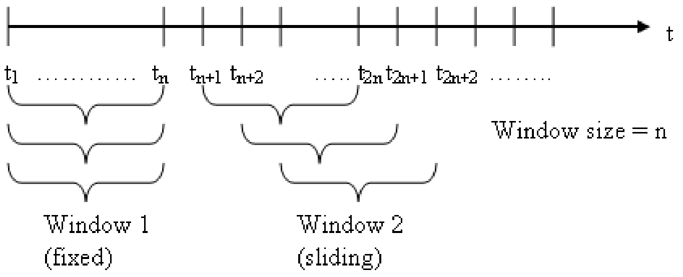
Investigating the changes in distribution that occur over time within the financial time series data and including them in the learning process is an ongoing research direction [9] [10] [12]. The benefits of the research in this area are not only algorithms that are adjusted to cope with the time element present in the data, but also algorithms that run online and can process data streams as well [15].

3 Distribution Based Samples Removing algorithm

The notion of examining the nature of distribution change in a time series and using it to filter the data samples is inspired by the technique of load shedding [22] using sliding windows. In order to develop an algorithm that can filter based on distribution change, we will first need to decide on an appropriate statistic for measuring differences in distribution.

We choose to use the Wilcoxon rank sum method (WXN) [13], which is a non-parametric test that assesses whether two sets of data samples follow the same distribution. It is easy to implement, efficient and a well known statistical test. We adopt the WXN method and use the change points it detects. The WXN paradigm is as follows: we set a fixed window on *n* points, [1,n], and starting after it, a sliding window of *n* points as well, [n+1, 2n], as shown in Figure 1. We move the second window and compare if the samples in both windows follow same distribution: if that is the case, we continue moving the second window, until the distribution changes. The change point will be at the last sample of the second window (point 2n+k); we move the first window just after that point [2n+k+1, 3n+k], the second window comes after the first one [3n+k+1, 4n+k] and we repeat the process for the rest of the dataset.

Figure 1: The Wilcoxon method with fixed reference window

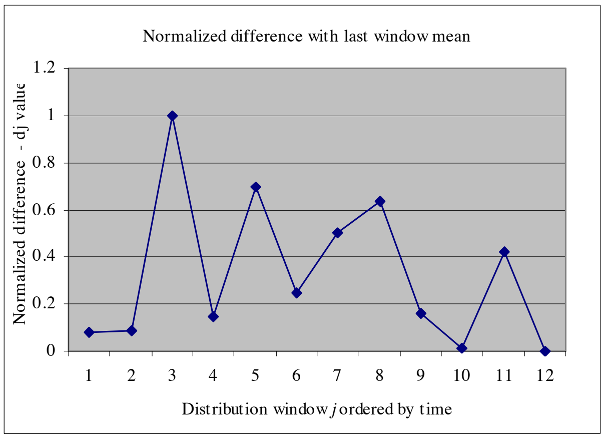

t
t_1 .... t_n t_{n+1} t_{n+2} ..... t_{2n} t_{2n+1} t_{2n+2} ....
Window size = n
Window 1
(fixed)
Window 2
(sliding)

After the WXN method has detected all the distribution change points in the training set of the time series, the mean (average) value of each window is calculated and compared to the mean value of the last (most recent in time) window: the difference between the mean value for a given window with index *j*, and the mean value of the last window, called Δavg[*j*,*last*]=mean$_j$-mean$_{last}$ is calculated, all differences are then normalized into the range of [0,1], giving us the value for d_j=Δavg[*j*,*last*]$_{Normalized}$.

$$d_j = \Delta avg[j,last]_{Normalized} = abs(\frac{\Delta avg[j,last]}{\max_i(\Delta avg[i,last])}) \qquad (1)$$

To gain an idea about likely behavior, we ran the WXN method on several real life time series (described in detail later in the paper) and the results showed the general pattern of Figure 2: some samples in the distant past were more similar to the most recent window than were some samples in the more recent past. We can see from Figure 2 moving left to right, there are windows in the most distant past with very similar distribution (windows 1 and 2) to the last window, and also windows in the not so distant past with quite different distribution (window 8) to the distribution of the last window. This confirmed our belief that many real time series are non-stationary, and that it is potentially promising to investigate methods for the filtering of samples based on similarity of distribution.

Figure 2: Example of $d_j = \Delta\text{avg}[j, last]_{Normalized}$ value between the distribution windows



We develop two versions of a **Distribution Based Samples Removing (DBSR) Algorithm**, one parametric and the other non-parametric. They both use information about the distribution changes in the time series for making the decision about which samples of the dataset to remove.

3.1 Distance value – threshold based decision

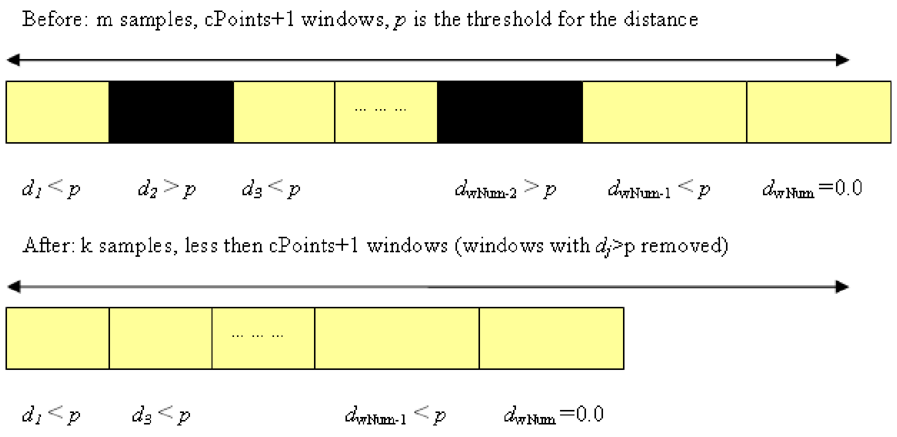
The parametric based DBSR (P-DBSR) algorithm requires the user to analyze the distribution change data: the size of the windows and the value of the distance to the most recent window. It requires a threshold value, between 0 and 1, and removes the samples from the windows where the distance to the most recent window is above the threshold value. The structure of the P-DBSR algorithm is as follows:

Algorithm P-DBSR

```
        Stage 1: change point detection
1:      Input: time series dataset $X = \{x_i \mid i = 1..m\}$
2:      Output: reduced time series $X = \{x_i \mid i = 1..k, k<m\}$
3:      Initial: reference windows $W_1$ and $W_2$, window size $n$, threshold value $p$,
        $W_1=\{x_1, ..x_n\}$, $W_2=\{x_{n+1},..,X_{2n}\}$, number of change points cPoints=0, distance
        values $d_j=\Delta\text{avg}[j,last]_{Normalized}$.
4:      While not the end of dataset
5:          Compare distribution for $W_1$ and $W_2$
6:      If $W_1$ and $W_2$ from the same distribution
7:          Move $W_2$ one sample forward
8:      Else
9:          Detect change point, cPoints += 1
10:         Set $W_1$ to start after $W_2$, then $W_2$ after $W_1$
11:     EndIf
12:     EndWhile
```

```
    Stage 2: parameter based dataset reduction
13: Calculate normalized distance values to the last window $d_j$, j=1.. cPoints+1
14: For all windows
15:     If window distance $d_j$ > p value
16:         Remove the current window
17: EndIf
18: EndFor
19: Return reduced dataset X = {$x_i$ | i = 1..k, k<m }
```

Figure 3: The parametric DBSR datasets, before and after removing the windows


Before: m samples, cPoints+1 windows, p is the threshold for the distance
... ...
$d_1 < p$
$d_2 > p$
$d_3 < p$
$d_{wNum-2} > p$
$d_{wNum-1} < p$
$d_{wNum} = 0.0$
After: k samples, less then cPoints+1 windows (windows with d_j>p removed)
... ...
$d_1 < p$
$d_3 < p$
$d_{wNum-1} < p$
$d_{wNum} = 0.0$

This version of the algorithm has several advantages: the user has access to the detailed information about the distribution, and can see how it changes over time, therefore getting insight into the volatility of the samples that will be used for forecasting; it will also indicate regions where the data may be noisy, and thus beneficial to remove.

We choose such value for *p* that would result in an amount is large enough for us to expect the final regression to be significantly different. Shown in Figure 3, the samples where the normalized distance was greater than the *p* value are in the black sections, and are removed at the end of the algorithm.

We assessed the algorithm over a range of values for the threshold - between 0.3 and 0.8. Some datasets had many windows with distributions similar to that of the last window, and in order to remove a significant amount of samples (around 30-35 %), those datasets required the threshold value set low. The datasets where there were windows with distribution quite different from the one of the last windows needed a threshold value set usually around 0.7 to remove the same percentage (30-35%) of samples. Even though the value for *p* was different for each dataset, the amount of samples removed was roughly the same for all datasets. We did so as we prefer to have same ratio of before and after dataset size, in order to test if removing such large amount of samples would be beneficial, regardless off the dataset.

3.2 Distance value – percentage based decision

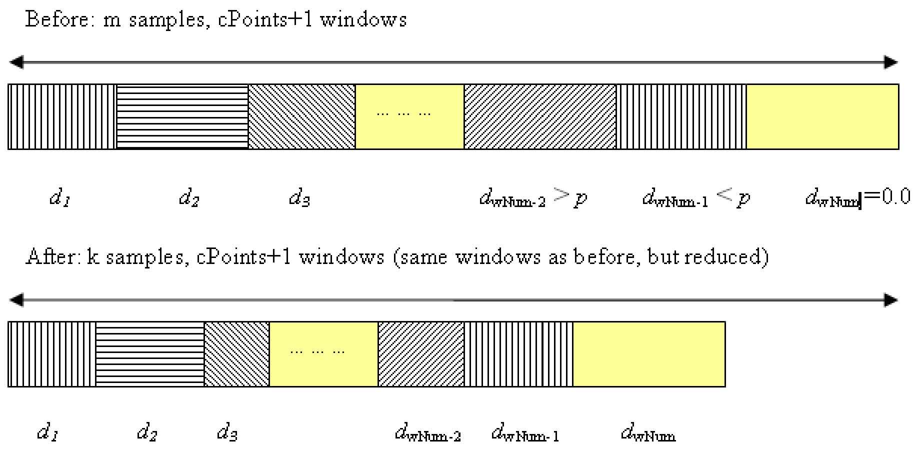
Our non-parametric DBSR (NP-DBSR) algorithm again accesses information about the distribution change and distribution distance with respect to the most recent window. As the distance is normalized in the range of 0-1, the algorithm uses that value to determine the portion of the window to be removed – e.g. if the normalized distance value for a given window is 0.7, the algorithm will remove 70% of the samples of that window. In other words, the samples from each window are filtered in proportion to the amount of their dissimilarity to the last window. This gives windows with a moderate value (moderate dissimilarity) for the distance some chance to contribute samples. Since distances are normalized, it will result in the most distant window having all of its instances removed, and the most recent window having no instances removed. Shown in Figure 4, we can see we have the same windows with different distributions (marked with different patterns) before and after, with the windows after being smaller, as the have samples being removed from them.

The structure of the NP-DBSR algorithm is as follows:

```
        Algorithm NP-DBSR

        Stage 1: change point detection (lines 1 - 12)
        Stage 2: parameter free dataset reduction
13:     Calculate normalized distance values to the last window $d$
$d_j = \Delta avg[j, last]_{Normalized \cdot j}$, j=1.. cPoints+1
14:     For all windows
15:         Remove $d_j$ *100 percent of the samples of the current window
16:     EndFor
17:     Return reduced dataset $X = \{x_i \mid i = 1..k, k<m\}$
```

Figure 4: Non parametric DBSR, before and after reducing the windows



4 Datasets

Our research was focused on forecasting stock market prices, as they are continuous series that can change very quickly, and are of great interest to both investors and researchers. We tested stock market prices of 12 random companies, with each dataset containing between 290 and 700 samples, recorded daily from a randomly chosen period between 1997 and 2010 [17][19]. We also tested a simulated dataset, where there did not exist many changes in the distribution, as well as the S&P quarterly index time series [18]. The stock market datasets were divided into a training and testing set, in the ratio of 9:1. We only focused on short term forecasting, so that the learning time of the machine learning models was small. The names of the companies, along with the number of samples and windows (changes) detected are listed in Table 1.

Since our technique focuses on data pre-processing, it can be used in conjunction with a large class of existing algorithms for time series prediction. We evaluated the use of our technique in conjunction with Linear Regression (LR), Pace Regression (PR), Support Vector machines (SVM) and Multilayer Perceptron (MLP). We did not evaluate the use of the popular ARIMA model, since that required an incompatible dataset format. We used the WEKA [19] software to run our experiments.

Table 1: Datasets used in the experiments

ID	Name	Samples/ windows	ID	Name	Samples/ windows (changes)
1	Amazon.com	422/13	8	Hewlett-Packard	612/27
2	Apple Computer	461/23	9	IBM	309/16
3	American Express	415/20	10	Island Pacific, Inc.	520/17
4	British Airways (ADS)	260/8	11	Johnson & Johnson	406/16
5	Colgate-Palmolive Co.	462/25	12	Simulated Dataset	475/22
6	eBay Inc.	520/22	13	S&P Quarterly Index	323/17
7	FedEx	423/17	14	Walt Disney Company	428/13

We used the five day relative difference in percentage of price (RDP) format [16]. The attributes by which the forecasted value was calculated were the 5, 10, 15 and 20 past days difference in percentage(RDP-5, RDP-10, RDP-15 and RDP-20), as well as a 15 day exponential moving average(EMA15). This type of transformation makes the data more symmetrical and closer to a normal distribution. The formulas that describe the RDP data format are listed in Table 2.

Table 2: RDP data format - attributes and forecast output

Input variables		Output variable	
EMA15	p(t)-EMA_{15}(t)	RDP+5	$(\overline{p(i+5)} - \overline{p(i)}) / \overline{p(i)} * 100$ $\overline{p(i)} = EMA_3(i)$
RDP-5	(p(t)-p(t-5))/p(t-5)*100		
RDP-10	(p(t)-p(t-10))/p(t-10) *100		
RDP-15	(p(t)-p(t-15))/p(t-15)*100		
RDP-20	(p(t)-p(t-20))/p(t-20)*100		

5 Experiments

The performances of the two versions of the algorithm were evaluated through the root mean square error (RMSE) metric. The results presented in Table 3 show the change in the RMSE value as captured in the formula:

(DBSR reduced dataset RMSE value) / (Full dataset RMSE value) * 100,

for both versions of the algorithm. i.e. The relative error using the filtered time series compared to using the full time series. In many cases for the machine learning methods, both versions of datasets filtered with our algorithms performed better than the machine learning methods trained on the full dataset, and in virtually all of them, employing at least one version of the algorithm resulted in a RMSE smaller than the methods trained on the full dataset.

The parametric method often yielded a smaller RMSE than the non-parametric method. The results in Table 3 also highlight some stability properties of the learning methods. As we can see from the RMSE reductions, the Linear Regression, Pace Regression and Support Vector Machines performed very similar when trained on the full datasets and on the reduced datasets as well, while Multilayer Perceptron performed poorly when trained on the full dataset, but had quite an improvement in performance when trained on some datasets filtered by the DBSR algorithm, but also had a large decrease in other cases.

Table 3: DBSMR algorithm change in RMSE values. Performances show percentage of RMSE error using our filtering approach compared to error without our filtering approach. Lower numbers indicate better performance for our filtering approach.

ID	P-DBSR *p*-val	LR P-DBSR	LR NP-DBSR	PR P-DBSR	PR NP-DBSR	SVM P-DBSR	SVM NP-DBSR	MLP P-DBSR	MLP NP-DBSR
1	0.5	98.30	95.72	98.07	93.82	100.2	96.56	61.32	64.04
2	0.3	92.53	95.11	91.86	95.49	86.98	96.87	114.6	103.6
3	0.4	92.06	83.91	97.35	85.34	103.6	94.82	120.4	106.6
4	0.7	94.95	99.93	97.28	99.55	97.19	96.89	84.51	102.3
5	0.8	98.72	101.1	98.12	98.89	98.52	99.37	105.3	98.56
6	0.6	99.32	104.7	100.1	103.4	100.9	107.1	99.00	126.9
7	0.6	98.77	98.51	97.77	99.65	95.71	99.86	181.3	105.8
8	0.5	95.77	97.38	95.72	96.37	99.14	100.2	60.59	91.52
9	0.7	95.22	97.80	98.44	97.07	94.31	99.93	100.6	86.12
10	0.5	100.9	99.27	102.9	99.69	96.90	95.73	78.31	83.74
11	0.7	91.66	97.82	94.37	100.6	95.05	103.3	95.54	95.68
12	0.5	96.81	94.57	96.78	94.53	98.53	95.22	84.92	107.1
13	0.5	93.40	93.98	91.75	90.96	90.75	125.8	158.8	139.6
14	0.5	98.70	104.2	98.97	102.4	97.23	102.8	81.08	109.6

6 Conclusion

Samples in financial time series datasets can be from different distributions and this creates challenges and opportunities for forecasting. We have developed data filtering algorithms that assess the importance of samples from a time series and retain those with most similarity to the recent past. Our experimental results show that the distribution of the data is indeed an important factor to consider, as we achieved reductions in forecasting error for time series with both few and many changes in the distribution. We believe our proposed DBSR algorithm is a simple and promising way to employ information about the distribution in the learning and prediction process.

In the future, we plan to investigate alternative methods to the Wilcoxon test for detecting distribution change and also investigate methods for stronger coupling of the distribution detection and prediction stages.

References

1. Tsay, R. S.: Analysis of Financial Time Series, Wiley-Interscience, 2005
2. Chatfield, C: The Analysis of Time Series: an Introduction, Chapman & Hall/CRC, 2004
3. Witten, I.H., Frank, E: Data mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2005
4. Alpaydin, E.: Introduction to Machine Learning, The MIT Press, 2004
5. Gaber, M. M., Zaslavsky A. and Krishnaswamy S.: Mining Data Streams: A Review, SIGMOD Record, 24(2), pp 18-26, 2005
6. Xindong, W., P. S. Yu, et al.: Data Mining: How Research Meets Practical Development?, Knowledge and Information Systems, 5(2), pp. 248-261, 2003.
7. Yoo, P. D., M. H. Kim, et al.: Machine Learning Techniques and Use of Event Information for Stock Market Prediction: A Survey and Evaluation, CIMCA-IAWTIC, 2005
8. Hulten, G., L. Spencer, et al.: Mining time-changing data streams, Proceedings of the seventh ACM SIGKDD international conference on Knowledge Discovery and Data Mining, pp. 97-106, San Francisco, California, 2001
9. Guozhu Dong, Jiawei Han, et al.: Online mining of changes from data streams: Research problems and preliminary results, in Proceedings of the 2003 ACM SIGMOD Workshop on Management and Processing of Data Streams, 2003
10. Chen, J., Gupta A. K.: Testing and locating variance changepoints with application to stock prices, Journal of the American Statistical Association 92(438), pp. 739-747, 1997
11. Adya, M. Collopy F.: How effective are neural networks at forecasting and prediction? A review and evaluation, Journal of Forecasting 17(5-6), pp. 481-495, 1998
12. Kifer, D., S. Ben-David, et al.: Detecting change in data streams, Proceedings of the Thirtieth international conference on Very large data bases,Toronto, Canada, VLDB Endowment: Volume 30, pp. 180-191, 2004
13. Hollander, M. and D. Wolfe: Nonparametric Statistical Methods, 2nd Edition, Wiley-Interscience, 1999
14. Kecman, V.: Learning and Soft Computing : support vector machines, neural networks, and fuzzy logic models, MIT Press, 2001
15. Xiaoyan Liu, Rui Zhang, et al.: Incremental Detection of Distribution Change in Stock Order Streams, 26th International Conference on Data Engineering Conference (ICDE). Long Beach, California, USA, 2010
16. Thomason M.: The Practitioner Methods and Tools, Journal of Computational Intelligence in Finance, 7(3), pp. 36-45, 1999
17. Web enabled scientific services and applications, http://www.wessa.net/stocksdata.wasp
18. Rob J. Hyndman S&P quarterly index online database, http://robjhyndman.com/tsdldata/data/9-17b.dat
19. Tsay, R. S.: Analysis of Financial Time Series datasets, http://faculty.chicagobooth.edu/ruey.tsay/teaching/fts/d-ibmln.dat
20. Waikato Environment for Knowledge Analysis (WEKA), http://www.cs.waikato.ac.nz/ml/weka/
21. Ganti, V., Gehrke, J., Ramakrishnan, R.: DEMON: mining and monitoring evolving data, IEEE Transactions on Knowledge and Data Engineering, Volume 13, Issue 1, 2001.
22. Babcock B., Datar M., Motwani R.: Load Shedding in Data Stream Systems, Proc. of the 2003 Workshop on Management and Processing of Data Streams (MPDS), 2003